RE

Public reporting burden for t
needed, and reviewing the c
Headquarters Service, Direc
Management and Budget, W

AD-A260 613

I PAGE

Form Approved
OPM No. 0704-0188

(2)

ng the time for reviewing instructions, searching existing data sources gathering and maintaining the data
other aspect of this collection of information, including suggestions for reducing this burden, to Washington
Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of

1. AGENCY USE ON

3. REPORT TYPE AND DATES COVERED
Final:18 Nov 92

4. TITLE AND SUBTITLE
Validation Summary Report: Alsys Limited, Alsys Limited, AlsyCOMP>062, Version 5.35, HP 9000 Series 8000 Model 827, (Host & Target), 921118N1.11298

5. FUNDING NUMBERS

6. AUTHOR(S)
National Computing Centre Limited
Manchester, UNITED KINGDOM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Computing Centre Limited
Oxford Road
Manchester MI 7ED
UNITED KINGDOM

8. PERFORMING ORGANIZATION REPORT NUMBER
AVF Control Number
90502/84-921214

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ada Joint Program Office
United States Department of Defense
Washington, D.C. 20301-3081

DTIC
ELECTE
FEB 08 1993
S E D

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Alsys Limited, Alsys Limited, AlsyCOMP>062, Version 5.35, HP 9000 Series 8000 Model 827, (Host & Target), ACVC 1.11.

14. SUBJECT TERMS
Ada programming language, Ada Compiler Val. Summary Report, Ada Compiler Val. Capability, Val. Testing, Ada Val. Office, Ada Val. Facility, ANSI/MIL-STD-1815A, AJPO.

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFED	UNCLASSIFIED	

NSN 7540-01-280-550

Standard Form 298, (Rev. 2-89)
Prescribed by ANSI Std. 239-128

AVF Control Number: 90502/84-921214

Ada COMPILER
VALIDATION SUMMARY REPORT:
Certificate Number: #921118N1.11298
Alsys Limited
AlsyCOMP_062 Version 5.35
HP 9000 Series 800 Model 827

Prepared By:
Testing Services
The National Computing Centre Limited
Oxford Road
Manchester
M1 7ED
England

Template Version 91-05-08

Certificate Information

The following Ada implementation was tested and determined to pass ACVC 1.11. Testing was completed on 12 November 1992.

Compiler Name and Version:	XD Ada MC68040/ARTX Version 1.2
Host Computer System:	Local Area VAX cluster (comprising VAXserver 3600, MicroVAX 2000 (2) and MicroVAX II machines) (under VMS 5.5).
Target Computer System:	Motorola MVME167 (MC68040) (bare machine)

See section 3.1 for any additional information about the testing environment.

As a result of this validation effort, Validation Certificate #921112N1.11197 is awarded to EDS-Scicon Defence Limited. This certificate expires 2 years after ANSI/MIL-STD-1815B is approved by ANSI.

This report has been reviewed and is approved.

Jon Leigh
Manager, System Software Testing
The National Computing Centre Limited
Oxford Road
Manchester
M1 7ED
England

for Ada Validation Organization
Director, Computer and Software Engineering Division
Institute for Defense Analyses
Alexandria VA 22311

Ada Joint Program Office
Dr. John Solomond, Director
Department of Defense
Washington DC 20301

DECLARATION OF CONFORMANCE

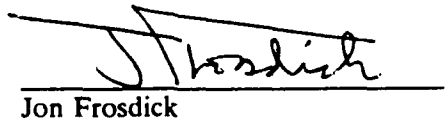
The following declaration of conformance was supplied by the customer.

Declaration of Conformance

Customer:	Alsys Limited
Ada Validation Facility:	The National Computing Centre Limited
ACVC Version:	1.11
Ada Implementation	
Ada Compiler Name and Version:	AlsyCOMP_062 Version 5.35
Host Computer System:	HP 9000 Series 800 Model 827 (under HP-UX Version 8.02)
Target Computer System:	HP 9000 Series 800 Model 827 (under HP-UX Version 8.02)

Declaration:

I, the undersigned, representing Alsys Limited, declare that Alsys Limited has no knowledge of deliberate deviations from the Ada Language Standard ANSI/MIL-STD-1815A, ISO 8652-1987, FIPS 119 as tested in this validation and documented in the Validation Summary Report.



Jon Frosdick
Director of Engineering
Alsys Limited

19/11/92
Date

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The Ada implementation described above was tested according to the Ada Validation Procedures [Pro92] against the Ada Standard [Ada83] using the current Ada Compiler Validation Capability (ACVC). This Validation Summary Report (VSR) gives an account of the testing of this Ada implementation. For any technical terms used in this report, the reader is referred to [Pro92]. A detailed description of the ACVC may be found in the current ACVC User's Guide [UG89].

1.1 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the Ada Certification Body may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

The organizations represented on the signature page of this report do not represent or warrant that all statements set forth in this report are accurate and complete, or that the subject implementation has no nonconformities to the Ada Standard other than those presented. Copies of this report are available to the public from the AVF which performed this validation or from:

National Technical Information Service
5285 Port Royal Road
Springfield VA 22161

Questions regarding this report or the validation test results should be directed to the AVF which performed this validation or to:

Ada Validation Organization
Computer and Software Engineering Division
Institute for Defense Analyses
1801 North Beauregard Street
Alexandria VA 22311-1772

1.2 REFERENCES

[Ada83] Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, February 1983 and ISO 8652-1987.

[Pro92] Ada Compiler Validation Procedures, Version 3.1, Ada Joint Program Office, August 1992.

[UG89] Ada Compiler Validation Capability User's Guide, 21 June 1989.

1.3 ACVC TEST CLASSES

Compliance of Ada implementations is tested by means of the ACVC. The ACVC contains a collection of test programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable. Class B and class L tests are expected to produce errors at compile time and link time, respectively.

The executable tests are written in a self-checking manner and produce a PASSED, FAILED, or NOT APPLICABLE message indicating the result when they are executed. Three Ada library units, the packages REPORT and SPPRT13, and the procedure CHECK_FILE are used for this purpose. The package REPORT also provides a set of identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The package SPPRT13 is used by many tests for Chapter 13 of the Ada Standard. The procedure CHECK_FILE is used to check the contents of text files written by some of the Class C tests for Chapter 14 of the Ada Standard. The operation of REPORT and CHECK_FILE is checked by a set of executable tests. If these units are not operating correctly, validation testing is discontinued.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that all violations of the Ada Standard are detected. Some of the class B tests contain legal Ada code which must not be flagged illegal by the compiler. This behaviour is also verified.

Class L tests check that an Ada implementation correctly detects violation of the Ada Standard involving multiple, separately compiled units. Errors are expected at link time, and execution is attempted.

In some tests of the ACVC, certain macro strings have to be replaced by implementation-specific values -- for example, the largest integer. A list of the values used for this implementation is provided in Appendix A. In addition to these anticipated test modifications, additional changes may be required to remove unforeseen conflicts between the tests and implementation-dependent characteristics. The modifications required for this implementation are described in section 2.3.

For each Ada implementation, a customized test suite is produced by the AVF. This customization consists of making the modifications described in the preceding paragraph, removing withdrawn tests (see section 2.1), and possibly removing some inapplicable tests (see section 2.2 and [UG89]).

In order to pass an ACVC an Ada implementation must process each test of the customized test suite according to the Ada Standard.

1.4 DEFINITION OF TERMS

Ada Compiler	The software and any needed hardware that have to be added to a given host and target computer system to allow transformation of Ada programs into executable form and execution thereof.
Ada Compiler Validation Capability (ACVC)	The means for testing compliance of Ada implementations, consisting of the test suite, the support programs, the ACVC user's guide and the template for the validation summary report.
Ada Implementation	An Ada compiler with its host computer system and its target computer system.
Ada Joint Program Office (AJPO)	The part of the certification body which provides policy and guidance for the Ada certification system.
Ada Validation Facility (AVF)	The part of the certification body which carries out the procedures required to establish the compliance of an Ada implementation.
Ada Validation Organization (AVO)	The part of the certification body that provides technical guidance for operations of the Ada certification system.
Compliance of an Ada Implementation	The ability of the implementation to pass an ACVC version.
Computer System	A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for the execution of the program; executes user-written or user-designated programs; performs user-designated date manipulation, including arithmetic operations and logic operations; and that can execute programs that modify themselves during execution. A computer system may be a stand-alone unit or may consist of several inter-connected units.
Conformity	Fulfilment of a product, process or service of all requirements specified.
Customer	An individual or corporate entity who enters into an agreement with an AVF which specifies the terms and conditions for AVF services (of any kind) to be performed.
Declaration of Conformance	A formal statement from a customer assuring that conformity is realized or attainable on the Ada implementation for which validation status is realized.
Host Computer System	A computer system where Ada source programs are transformed into executable form.

Inapplicable test	A test that contains one or more test objectives found to be irrelevant for the given Ada implementation.
ISO	International Organization for Standardization.
LRM	The Ada standard, or Language Reference Manual, published as ANSI/MIL-STD-1815A-1983 AND ISO 8652-1987. Citations from the LRM take the form "<section>.<subsection>:<paragraph>."
Operating System	Software that controls the execution of programs and that provides services such as resource allocation, scheduling, input/output control and data management. Usually, operating systems are predominantly software, but partial or complete hardware implementations are possible.
Target Computer System	A computer system where the executable form of Ada programs are executed.
Validated Ada Compiler	The compiler of a validated Ada implementation.
Validated Ada Implementation	An Ada implementation that has been validated successfully either by AVF testing or by registration [Pro92].
Validation	The process of checking the conformity of an Ada compiler to the Ada programming language and of issuing a certificate for this implementation.
Withdrawn test	A test found to be incorrect and not used in conformity testing. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains erroneous or illegal use of the Ada programming language.

CHAPTER 2

IMPLEMENTATION DEPENDENCIES

2.1 WITHDRAWN TESTS

The following tests have been withdrawn by the AVO. The rationale for withdrawing each test is available from either the AVO or the AVF. The publication date for this list of withdrawn tests is 2 August 1991.

E28005C	B28006C	C32203A	C34006D	C35508I	C35508J
C35508M	C35508N	C35702A	C35702B	B41308B	C43004A
C45114A	C45346A	C45612A	C45612B	C45612C	C45651A
C46022A	B49008A	B49008B	A74006A	C74308A	B83022B
B83022H	B83025B	B83025D	C83026A	B83026B	C83041A
B85001L	C86001F	C94021A	C97116A	C98003B	BA2011A
CB7001A	CB7001B	CB7004A	CC1223A	BC1226A	CC1226B
BC3009B	BD1B02B	BD1B06A	AD1B08A	BD2A02A	CD2A21E
CD2A23E	CD2A32A	CD2A41A	CD2A41E	CD2A87A	CD2B15C
BD3006A	BD4008A	CD4022A	CD4022D	CD4024B	CD4024C
CD4024D	CD4031A	CD4051D	CD5111A	CD7004C	ED7005D
CD7005E	AD7006A	CD7006E	AD7201A	AD7201E	CD7204B
AD7206A	BD8002A	BD8004C	CD9005A	CD9005B	CDA201E
CE2107I	CE2117A	CE2117B	CE2119B	CE2205B	CE2405A
CE3111C	CE3116A	CE3118A	CE3411B	CE3412B	CE3607B
CE3607C	CE3607D	CE3812A	CE3814A	CE3902B	

2.2 INAPPLICABLE TESTS

A test is inapplicable if it contains test objectives which are irrelevant for a given Ada implementation. Reasons for a test's inapplicability may be supported by documents issued by the ISO and the AJPO known as Ada Commentaries and commonly referenced in the format AI-ddddd. For this implementation, the following tests were determined to be inapplicable for the reasons indicated; references to Ada Commentaries are included as appropriate.

The following 201 tests have floating-point type declarations requiring more digits than SYSTEM.MAX_DIGITS:

C24113L..Y (14 tests)	C35705L..Y (14 tests)
C35706L..Y (14 tests)	C35707L..Y (14 tests)
C35708L..Y (14 tests)	C35802L..Z (15 tests)
C45241L..Y (14 tests)	C45321L..Y (14 tests)
C45421L..Y (14 tests)	C45521L..Z (15 tests)
C45524L..Z (15 tests)	C45621L..Z (15 tests)
C45641L..Y (14 tests)	C46012L..Z (15 tests)

The following 20 tests check for the predefined type LONG_INTEGER; for this implementation, there is no such type:

C35404C	C45231C	C45304C	C45411C	C45412C
C45502C	C45503C	C45504C	C45504F	C45611C
C45613C	C45614C	C45631C	C45632C	B52004D
C55B07A	B55B09C	B86001W	C86006C	CD7101F

C35713B, C45423B, B86001T, and C86006H check for the predefined type SHORT_FLOAT; for this implementation, there is no such type.

C35713D and B86001Z check for a predefined floating-point type with a name other than FLOAT, LONG_FLOAT, or SHORT_FLOAT; for this implementation, there is no such type.

C45531M..P and C45532M..P (8 tests) check fixed-point operations for types that require a SYSTEM.MAX_MANTISSA of 47 or greater; for this implementation, MAX_MANTISSA is less than 47.

C45536A, C46013B, C46031B, C46033B, and C46034B contain length clauses that specify values for 'SMALL that are not powers of two or ten; this implementation does not support such values for 'SMALL.

C45624A..B (2 tests) check that the proper exception is raised if MACHINE_OVERFLOWS is FALSE for floating point types and the results of various floating-point operations lie outside the range of the base type; for this implementation, MACHINE_OVERFLOWS is TRUE.

B86001Y uses the name of a predefined fixed-point type other than type DURATION; for this implementation, there is no such type.

CD1009C checks whether a length clause can specify a non-default size for a floating-point type; this implementation does not support such sizes.

CD2A53A checks operations of a fixed-point type for which a length clause specifies a power-of-ten TYPE'SMALL; this implementation does not support decimal 'SMALLs. (See section 2.3.)

CD2A84A, CD2A84E, CD2A84I..J (2 tests), and CD2A84O use length clauses to specify non-default sizes for access types; this implementation does not support such sizes.

BD8001A, BD8003A, BD8004A..B (2 tests), and AD8011A use machine code insertions; this implementation provides no package MACHINE_CODE.

The tests listed in the following table check that USE_ERROR is raised if the given file operations are not supported for the given combination of mode and access method; this implementation supports these operations.

Test	File Operation	Mode	File Access Method
CE2102E	CREATE	OUT_FILE	SEQUENTIAL_IO
CE2102F	CREATE	INOUT_FILE	DIRECT_IO
CE2102J	CREATE	OUT_FILE	DIRECT_IO
CE2102N	OPEN	IN_FILE	SEQUENTIAL_IO
CE2102O	RESET	IN_FILE	SEQUENTIAL_IO
CE2102P	OPEN	OUT_FILE	SEQUENTIAL_IO
CE2102Q	RESET	OUT_FILE	SEQUENTIAL_IO
CE2102R	OPEN	INOUT_FILE	DIRECT_IO
CE2102S	RESET	INOUT_FILE	DIRECT_IO
CE2102T	OPEN	IN_FILE	DIRECT_IO
CE2102U	RESET	IN_FILE	DIRECT_IO
CE2102V	OPEN	OUT_FILE	DIRECT_IO
CE2102W	RESET	OUT_FILE	DIRECT_IO
CE3102F	RESET	Any Mode	TEXT_IO
CE3102G	DELETE	--------	TEXT_IO
CE3102I	CREATE	OUT_FILE	TEXT_IO
CE3102J	OPEN	IN_FILE	TEXT_IO
CE3102K	OPEN	OUT_FILE	TEXT_IO

The tests listed in the following table check the given file operations for the given combination of mode and access method; this implementation does not support these operations.

Test	File Operation	Mode	File Access Method
CE2105A	CREATE	IN_FILE	SEQUENTIAL_IO
CE2105B	CREATE	IN_FILE	DIRECT_IO
CE3109A	CREATE	IN_FILE	TEXT_IO

CE2203A checks that WRITE raises USE_ERROR if the capacity of an external sequential file is exceeded; this implementation cannot restrict file capacity.

EE2401D, EE2401G and CE2401H use instantiations of DIRECT_IO with unconstrained array and record types; this implementation raises USE_ERROR on the attempt to create a file of such types.

CE2403A checks that WRITE raises USE_ERROR if the capacity of an external direct file is exceeded; this implementation cannot restrict file capacity.

CE3304A checks that SET_LINE_LENGTH and SET_PAGE_LENGTH raise USE_ERROR if they specify an inappropriate value for the external file; there are no inappropriate values for this implementation.

CE3413B checks that PAGE raises LAYOUT_ERROR when the value of the page number exceeds COUNT'LAST; for this implementation, the value of COUNT'LAST is greater than 150000, making the checking of this objective impractical.

CE3202A expects that function NAME can be applied to the standard input and output files; in this implementation these files have no names, and USE_ERROR is raised. (See section 2.3.)

2.3 TEST MODIFICATIONS

Modifications (see section 1.3) were required for 26 tests.

The following tests were split into two or more tests because this implementation did not report the violations of the Ada Standard in the way expected by the original tests.

B23004A	B24007A	B24009A	B28003A	B32202A
B32202B	B32202C	B37004A	B61012A	B74304A
B74401F	B74401R	B91004A	B95032A	B95069A
B95069B	B97103E	BA1101B2	BA1101B4	BC2001D
BC3009C				

B85002A was graded passed by Evaluation Modification as directed by the AVO. This test declares a record type REC2 whose sole component is of an unconstrained record type with a size in excess of 2**32 bytes; this implementation rejects the declaration of REC2. Although a strict interpretation of the LRM requires that this type declarations be accepted (an exception may be raised on the elaboration of the type or an object declaration), *the AVO accepted this behaviour in consideration* that such early error detection is expected to be allowed by the revised language standard.

BA2001E was graded passed by Evaluation Modification as directed by the AVO. The test expects that duplicate names of subunits with a common ancestor will be detected as compilation errors; this implementation detects the errors at link time, and the AVO ruled that this behaviour is acceptable.

EA3004D was graded passed by Evaluation and Processing Modification as directed by the AVO. The test requires that either pragma INLINE is obeyed for a function call in each of three contexts and that thus three library units are made obsolete by the re-compilation of the inlined function's body, or else the pragma is ignored completely. This implementation obeys the pragma except when the call is within the package specification. When the test's files are processed in the given order, only two units are made obsolete; thus, the expected error at line 27 of file EA3004D6M is not valid and is not flagged. To confirm that indeed the pragma is not obeyed in this one case, the test was also processed with the files re-ordered so that the re-compilation follows only the package declaration (and thus the other library units will not be made obsolete, as they are compiled later); a "NOT APPLICABLE" result was produced, as expected. The revised order of files was 0-1-4-5-2-3-6.

CD2A53A was graded inapplicable by Evaluation Modification as directed by the AVO. The test contains a specification of a power-of-10 value as 'SMALL for a fixed-point type. The AVO ruled that, under ACVC 1.11, support of decimal 'SMALLs may be omitted.

CE3202A was graded inapplicable by Evaluation Modification as directed by the AVO. This test applies function NAME to the standard input file, which in this implementation has no name; USE_ERROR is raised but not handled, so the test is aborted. The AVO ruled that this behaviour is acceptable pending any resolution of the issue by the ARG.

CHAPTER 3

PROCESSING INFORMATION

3.1 TESTING ENVIRONMENT

The Ada implementation tested in this validation effort is described adequately by the information given in the initial pages of this report, together with the following:

The memory size of the Host/Target Configuration is 64 Mbytes.

For technical information about this Ada implementation, contact:

Con Bradley
Alsys Limited
Partridge House
Newtown Road
Henley-on-Thames
Oxfordshire
RG9 1EH

For sales information about this Ada implementation, contact:

Pascal Plisson
Alsys SA
29 Avenue Rue Lucien Rene Duchesne
La Chataigneraie
78170 La Celle St Cloud
France

Testing of this Ada implementation was conducted at the customer's site by a validation team from the AVF.

3.2 SUMMARY OF TEST RESULTS

An Ada Implementation passes a given ACVC version if it processes each test of the customized test suite in accordance with the Ada Programming Language Standard, whether the test is applicable or inapplicable; otherwise, the Ada Implementation fails the ACVC [Pro92].

For all processed tests (inapplicable and applicable), a result was obtained that conforms to the Ada Programming Language Standard.

The list of items below gives the number of ACVC tests in various categories. All tests were processed, except those that were withdrawn because of test errors (item b; see section 2.1), those that require a floating-point precision that exceeds the implementation's maximum precision (item e; see section 2.2), and those that depend on the support of a file system -- if none is supported (item d). All tests passed, except those that are listed in sections 2.1 and 2.2 (counted in items b and f, below).

a)	Total Number of Applicable Tests	3791
b)	Total Number of Withdrawn Tests	95
c)	Processed Inapplicable Tests	284
d)	Non-Processed I/O Tests	0
e)	Non-Processed Floating-Point Precision Tests	0
f)	Total Number of Inapplicable Tests	284
g)	Total Number of Tests for ACVC 1.11	4170 (a+b+f)

3.3 TEST EXECUTION

A tar format cartridge tape containing the customized test suite (see section 1.3) was taken on-site by the validation team for processing. The contents of the tar format cartridge tape was loaded on to an RDI Britelite machine running SUNOS 4.1.1. The ACVC was then copied onto a DAT tape in tar format using a HP710 (9000 series) machine via an NFS link. The DAT tape was then loaded directly onto the host computer.

After the test files were loaded onto the host computer, the full set of tests was processed by the Ada implementation.

Testing was performed using command scripts provided by the customer and reviewed by the validation team. See Appendix B for a complete listing of the processing options for this implementation. It also indicates the default options. The options invoked explicitly for validation testing during this test in order to produce full compilation listings were:

-e 999 Stop compilation after 999 errors

-B Produce a compilation listing.

Test output, compiler and linker listings, and job logs were captured on cartridge tape (tar format) and archived at the AVF. The listings examined on-site by the validation team were also archived.

APPENDIX A

MACRO PARAMETERS

This appendix contains the macro parameters used for customizing the ACVC. The meaning and purpose of these parameters are explained in [UG89]. The parameter values are presented in two tables. The first table lists the values that are defined in terms of the maximum input-line length, which is the value for $MAX_IN_LEN--also listed here. These values are expressed here as Ada string aggregates, where "V" represents the maximum input-line length.

Macro Parameter	Macro Value
$MAX_IN_LEN	255 -- Value of V
$BIG_ID1	(1..V-1 => 'A', V => '1')
$BIG_ID2	(1..V-1 => 'A', V => '2')
$BIG_ID3	(1..V/2 => 'A') & '3' & (1..V-1-V/2 => 'A')
$BIG_ID4	(1..V/2 => 'A') & '4' & (1..V-1-V/2 => 'A')
$BIG_INT_LIT	(1..V-3 => '0') & "298"
$BIG_REAL_LIT	(1..V-5 => '0') & "690.0"
$BIG_STRING1	'"' & (1..V/2 => 'A') & '"'
$BIG_STRING2	'"' & (1..V-1-V/2 => 'A') & '1' & '"'
$BLANKS	(1..V-20 => ' ')
$MAX_LEN_INT_BASED_LITERAL	"2:" & (1..V-5 => '0') & "11:"
$MAX_LEN_REAL_BASED_LITERAL	"16:" & (1..V-7 => '0') & "F.E:"
$MAX_STRING_LITERAL	'"' & (1..V-2 => 'A') & '"'

The following table lists all of the other macro parameters and their respective values.

Macro Parameter	Macro Value
$ACC_SIZE	32
$ALIGNMENT	4
$COUNT_LAST	2147483647
$DEFAULT_MEM_SIZE	2147483647
$DEFAULT_STOR_UNIT	8
$DEFAULT_SYS_NAME	HP9000_PA_RISC
$DELTA_DOC	2#1.0#E-31
$ENTRY_ADDRESS	ENTRY_ADDR
$ENTRY_ADDRESS1	ENTRY_ADDR1
$ENTRY_ADDRESS2	ENTRY_ADDR2
$FIELD_LAST	255
$FILE_TERMINATOR	' '
$FIXED_NAME	NO_SUCH_FIXED_TYPE
$FLOAT_NAME	NO_SUCH_FLOAT_TYPE
$FORM_STRING	""
$FORM_STRING2	"CANNOT_RESTRICT_FILE_CAPACITY"
$GREATER_THAN_DURATION	100_000.0
$GREATER_THAN_DURATION_BASE_LAST	100_000_000.0
$GREATER_THAN_FLOAT_BASE_LAST	1.80141E+38
$GREATER_THAN_FLOAT_SAFE_LARGE	1.0E38

$GREATER_THAN_SHORT_FLOAT_SAFE_LARGE
1.0E38

$HIGH_PRIORITY 16

$ILLEGAL_EXTERNAL_FILE_NAME1
*not_there//not_there/** ^

$ILLEGAL_EXTERNAL_FILE_NAME2
not_there/not_there/not_there/./../not_there///

$INAPPROPRIATE_LINE_LENGTH -1

$INAPPROPRIATE_PAGE_LENGTH
-1

$INCLUDE_PRAGMA1 PRAGMA INCLUDE ("A28006D1.TST")

$INCLUDE_PRAGMA2 PRAGMA INCLUDE ("B28006D1.TST")

$INTEGER_FIRST -2147483648

$INTEGER_LAST 2147483647

$INTEGER_LAST_PLUS_1 2147483648

$INTERFACE_LANGUAGE C

$LESS_THAN_DURATION -100_000.0

$LESS_THAN_DURATION_BASE_FIRST
-100_000_000.0

$LINE_TERMINATOR ' '

$LOW_PRIORITY 1

$MACHINE_CODE_STATEMENT NULL;

$MACHINE_CODE_TYPE NO_SUCH_TYPE

$MANTISSA_DOC 31

$MAX_DIGITS 15

$MAX_INT 2147483647

$MAX_INT_PLUS_1 2147483648

$MIN_INT	-2147483648
$NAME	SHORT_SHORT_INTEGER
$NAME_LIST	HP9000_PA_RISC
$NAME_SPECIFICATION1	/usr/users/alsys/ltd/qacommands/work/objll/ce/X2120A
$NAME_SPECIFICATION2	/usr/users/alsys/ltd/qacommands/work/objll/ce/X2120B
$NAME_SPECIFICATION3	/usr/users/alsys/ltd/qacommands/work/objll/ce/X3119A
$NEG_BASED_INT	16#FF_FF_FF_FD#
$NEW_MEM_SIZE	1048576
$NEW_STOR_UNIT	8
$NEW_SYS_NAME	HP9000_PA_RISC
$PAGE_TERMINATOR	ASCII.FF
$RECORD_DEFINITION	NEW INTEGER
$RECORD_NAME	NO_SUCH_MACHINE_CODE_TYPE
$TASK_SIZE	32
$TASK_STORAGE_SIZE	32768
$TICK	0.010
$VARIABLE_ADDRESS	VARIABLE ADDR
$VARIABLE_ADDRESS1	VARIABLE_ADDR1
$VARIABLE_ADDRESS2	VARIABLE_ADDR2
$YOUR_PRAGMA	EXPORT_OBJECT

APPENDIX B

COMPILATION SYSTEM OPTIONS

The compiler options of this Ada implementation, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this appendix are to compiler documentation and not to this report.

LINKER OPTIONS

The linker options of this Ada implementation, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this appendix are to linker documentation and not to this report.

APPENDIX C

APPENDIX F OF THE Ada STANDARD

The only allowed implementation dependencies correspond to implementation-dependent pragmas, to certain machine-dependent conventions as mentioned in Chapter 13 of the Ada Standard, and to certain allowed restrictions on representation clauses. The implementation-dependent characteristics of this Ada implementation, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this Appendix are to compiler documentation and not to this report. Implementation-specific portions of the package STANDARD, which are not a part of Appendix F, are:

```
package STANDARD is

...

type SHORT_SHORT_INTEGER is range -128 .. 127;

type SHORT_INTEGER is range -32768 .. 32767;

type INTEGER is range -2147483648 .. 2147483647;

type FLOAT is digits 6 range -2#1.111_1111_1111_1111_1111_1111#E+127 ..
2#1.111_1111_1111_1111_1111_1111#E+127;

type LONG_FLOAT is digits 15 range

-2#1.1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111#E1023 ..

2#1.1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111_1111#E1023;

type DURATION is delta 2#0.00000000000001# range -86400.0 .. 86400.0;

...

end STANDARD;
```